This paper appeared at IEEE VTC Spring 2016.

Performance evaluation of TCP-based traffic over direct communications in LTE-Advanced

Giovanni Nardini, Giovanni Stea, Antonio Virdis

Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy
g.nardini@ing.unipi.it, giovanni.stea@unipi.it, a.virdis@iet.unipi.it

***Abstract*— Direct (or device-to-device, D2D) communications are being investigated in the framework of LTE-Advanced. They allow one-to-one communications between two endpoints, under the control of the eNodeB, which allocates resources for the d2d flow, but does not act as a relay for its traffic. The direct link can also be used for file transfer or proximity-based browsing, i.e. applications running on TCP. In this paper, we evaluate the performance of TCP-based traffic transported through the direct link, in several scenarios. We show and explain non-intuitive results, which arise from the interplay of TCP and LTE-A protocol mechanisms, and compare the existing TCP versions in a dynamic environment, where mode switches between the direct and the infrastructure link may induce periodic losses.**

***Index Terms*—LTE-A, device-to-device, TCP, performance evaluation**

I. INTRODUCTION

Network-controlled Device-to-device (D2D) communications are envisaged to abate latency and allow spatial frequency reuse. They are expected to support both broadcast *and* unicast services. The latter are expected to support several applications, such as file transfer and proximity-based file sharing and browsing. It is foreseeable that TCP will be used to support the latter, since i) existing applications rely on it for assured in-sequence delivery, and ii) its endpoint-regulated congestion control is useful in a network with shared resources such as LTE-A.

Previous works considered the interplay between TCP mechanisms and LTE-A ones, e.g. [22]-[24]. However, no work that we know of evaluates the performance of TCP-based traffic over a D2D link. In this paper, we show how the Round Trip Time (RTT) of a TCP connection is affected by D2D communications. We then show the effects of D2D *mode switch* on a TCP flow, since the ability of a D2D link to switch between direct and infrastructure mode has been widely studied by the research community [14]-[18]. To this aim, we compare the performance obtained by different TCP implementations, like Reno, NewReno and so on. Our evaluation is carried out using SimuLTE [2]-[3], a C++ system-level simulator developed for OMNeT++ [4], which simulates the data plane of the LTE/LTE-A radio access network, including the entire protocol stack from the PDCP to the physical layer, where we implemented one-to-one direct communications. We evaluate a *static* scenario, where flows are sent through either the *sidelink* (SL) or the *uplink/downlink* (UL/DL) infrastructure path for their entire lifetime, and a *dynamic* one, where flows can be switched between the two paths. We show that direct communications may reduce the RTT of a TCP connection, although not as much as expected due to interactions with LTE-A protocol mechanisms. Actually, in some cases, the UL/DL path may outperform the SL one. Moreover, we show that mode switching impairs the performance of TCP-based applications, as it causes losses which are interpreted by TCP as a congestion signal. As a result, the throughput is highly affected by the version of congestion control algorithm implemented in the TCP. We compare the most common algorithms and show why some faster than the others from mode switches.

The rest of the paper is organized as follows: Section II reports background. Section III analyzes the static scenario, whereas the dynamic one is discussed in Section IV. We conclude the paper in Section V.

II. BACKGROUND

Hereafter we provide a minimal background on LTE-A and introduce our working hypotheses. Going from top to bottom through the LTE-A stack, we find the Packet Data Convergence Protocol (PDCP), where IP packets are ciphered and numbered and immediately sent down to the Radio Link Control (RLC), where they are buffered. The MAC requests to the RLC an RLC PDU of a given size, and the RLC responds by dequeuing from its buffer an appropriate number of RLC SDUs, fragmenting and concatenating them as necessary to fit the request (the RLC *unacknowledged mode* (UM) is recommended by the standard for D2D [1]). MAC-layer transmissions are arranged in subframes and paced at Transmission Time Intervals (TTIs) of 1ms. In the downlink (DL), the eNB allocates a vector of *Resource Blocks* (RBs) to transmissions directed to the User Equipments (UEs) associated to it on each TTI. Each RB carries a number of bits depending on the Channel Quality Indicator (CQI) reported by the UE. MAC-level error recovery is provided by a Hybrid ARQ (H-ARQ) scheme, which allows a configurable number of retransmissions. Retransmission is *asynchronous* in the DL and *synchronous* in the UL, where it occurs after eight TTIs.

UEs access UL resources through a Random Access Procedure (RAC). RAC request collisions are resolved through backoff. RAC requests are responded by scheduling the UE in a future TTI, and are re-iterated if unanswered after a timeout. The handshake for UL transmissions takes five messages (Figure 1): first the UE sends a RAC request; the eNB responds with a short grant, large enough for a *Buffer Status Report* (BSR); the UE sends its BSR; the eNB sends a larger grant according to its scheduling policy, and the UE transmits its data. The middle two

interactions can be avoided if the UE sends PDUs *together* with the BSR, a technique known as *bandwidth stealing* (BS). Semi-Persistent Scheduling (SPS) can also be used to transmit *periodic* traffic, e.g., VoIP, and consists in the eNB issuing a long-term, periodic grant to a UE, which can then transmit in the pre-assigned RBs without signaling. However, SPS prevents link adaptation, hence it is inefficient [21].

For D2D communications to happen, the eNB issues two grants for *the same* RBs to *a* for transmission, and to *b* for reception, simultaneously (slots 5 and 3 of Figure 1 respectively). We consider a Frequency Division Duplexing (FDD) system, where DM transmissions take place in the UL subframe, which is less likely to be the loaded (due to the well-known traffic asymmetry) and allows better overall SINR [19]. Accordingly, we assume that UEs are equipped with a Single Carrier-Frequency Division Multiple Access (SC-FDMA) receiver [20]. As far as H-ARQ is concerned, we assume that the feedback is sent by the D2D receiver to *both* the sender and the eNB. This is necessary, since the former needs to know if retransmission is required (in eight TTI, it being UL), while the latter has to allocate RBs for it to take place. On the other hand, it cannot be given for granted that the eNB is able to overhear the *transmission* occurring on the SL, since the power at the sender may not be sufficient.

For several reasons, e.g. link quality changes due to user mobility, D2D must include a mechanism to switch a flow from the SL to the UL/DL path and vice versa dynamically. When this happens, *losses* may occur. In fact, the standard mandates that different LTE connections be used for the SL and the UL/DL. Now, different connections have unrelated ciphering and numbering. Therefore, traffic buffered at the RLC of one connection cannot be sent on the *other* connection after a mode switch, since its ciphering and numbering do not fit with the new one. Thus, the only option is to discard it and have it retransmitted at the application level.

A. Background on TCP

We briefly mention the main features of TCP. TCP provides reliable, duplicate-free and ordered delivery of application data. Once an end-to-end connection is established, TCP breaks the sending application's stream of bytes into a set of *segments*, each one identified by the sequence number of the last byte in it. The reception of a segment must be notified by sending back an *acknowledgment* (ACK), which contains the sequence number of the next expected byte, and confirms that all previous bytes have been correctly received. A *duplicate ACK* (dupACK) is sent by the receiver when out-of-sequence bytes arrive, which is likely to signal that one or more segments are missing. If the ACK is not received before a retransmission timer expires, the segment is retransmitted. The number of segments that can be sent simultaneously is limited by the *flow control* and *congestion control* mechanisms. Both use a sliding window to avoid sending more data than those the receiver and the network, respectively, can handle. The effective sending window is the minimum of the two. While the size of the *receive window*, which paces flow control, is specified by the receiver itself, the status of the network must be inferred from the (lack of) reception of ACKs. Most congestion control algorithms begin with a *slow-start* phase, where the congestion window (cwnd) is increased exponentially, i.e. doubled at each RTT. When a threshold is reached, the algorithm enters a congestion-avoidance phase, where the cwnd is increased linearly (cubically in TCP Cubic, [12]). If a retransmission timer expires, TCP assumes the network is congested, hence reduces the cwnd.

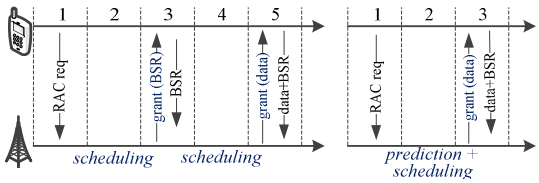

Figure 1 – Standard uplink scheduling (left) and bandwidth stealing (right).

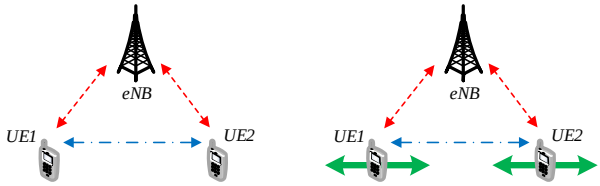

Figure 2 – Static scenario (left) and dynamic scenario (right).

TABLE 1 - SIMULATION PARAMETERS

Parameter	Value
Carrier frequency	2 GHz
Bandwidth	10 MHz (50 RBs)
Path loss model	Urban Macro [13]
eNB Tx Power	40 dBm
UE Tx Power	20 dBm
eNB Antenna gain	18 dB
Noise figure	5 dB
Cable loss	2 dB
RAC timeout	10ms
Simulation time	500 seconds
# of independent repetitions	10

III. STATIC D2D CONNECTIONS

In this section, we evaluate the Round Trip Time (RTT) and the throughput of a TCP connection between two static, D2D-capable UEs, communicating in DM and IM. A TCP connection is bidirectional, including a *data* flow, consisting of (long) TCP segments, and an *ACK* flow. The data and ACK flows are unrelated at the MAC level, hence they can be sent through either the DM or the IM path independently. Thus, we have four scenarios, corresponding to the cases of data/ACK flows routed as DM/DM, DM/IM, IM/DM, IM/IM.

Figure 2 (left) reports the simulation scenario. We consider one pair of D2D-capable UEs and one eNB, whose antenna radiates the signal with an omnidirectional pattern. The two UEs are 20m away from the eNB and 20m away from each other. This way, CQIs stay equal to 15 for the whole simulation for SL, UL and DL directions, hence measurements are not affected by factors like different link quality. Simulation parameters are reported in Table 1. UE1 sends a 1GB file to UE2. The sending rate is limited by the TCP receive window size, which is 8 KB. Figure 3 shows the evolution of the RTT. The RTT is about

42ms (on average) for traditional IM-IM communication, whereas DM-DM reduces it to 30ms, but not less. In other words, DM transmissions cannot drastically abate the RTT, unless SPS or similar mechanisms are employed. This can be explained by looking at Figure 4, which shows the sequence of events between the transmission of one TCP data segment and the reception of the corresponding TCP ACK, when DM-DM is used. At time *t*=0, the data segment is available in the transmission buffer of UE1. Since the latter has no scheduled resources, it issues a RAC request to the eNB, which replies with an UL grant of the minimum size (one RB) to allow UE1 to send the BSR, at *t*=5. The eNB decodes the BSR at *t*=9 and sends a SL grant to UE1, which uses it to transmit data to UE2 at *t*=11. At *t*=15, UE2 decodes the airframe and delivers it to the TCP layer. The TCP ACK goes back to UE1 following the same sequence, i.e. UE1 receives the ACK at *t*=30. Two thirds of the time are occupied by the RAC procedure and BSR reporting, which are unavoidable. For the IM-DM case, the RTT is longer, since the data flow has to traverse two hops: the UL leg requires the same timing as a DM transmission, while the DL leg adds 5ms.

When the *ACK* flow is in IM, instead, BS, as described in Section II, comes into play. Note that BS *cannot* work for DM, since it requires that the recipient of *both* BSR *and* data be the same entity (i.e., the eNB), whereas in DM data would be sent to the peering UE instead. Moreover, it seldom works for the *data* flow in IM, since TCP segments are too large to fit the one and only RB granted for the BSR after a successful RAC request. In fact, the RLC layer provides a RLC PDU of the requested size to the MAC layer. However, since TCP data segments are likely to be large, the RLC fragments them, thus only the first fragment fits the granted RB. Although that fragment reaches the eNB's RLC immediately, the original data segment is not reassembled until *all* the subsequent fragments have arrived at the eNB, hence BS is of little benefit. The ACK flow consists of small segments (46 bytes at the MAC level), which can instead be accommodated in that single RB granted to UE2. If a burst of ACKs is queued at UE2, one or two of them can be sent together with the BSR, while the others need to wait for a subsequent data grant. This results in irregular arrivals of TCP ACKs, hence irregular sending of new TCP data segments. This is visible in the DM-IM and IM-IM lines in Figure 3, where the RTT fluctuates. With reference to Figure 4, if new data becomes available at *t*+6, right after UE1 has sent its BSR to the eNB, UE1 *cannot* start a new RAC request, since it is already waiting for a data grant. At the same time, it is likely that the newly arrived data do not fit the grant decoded at *t*+9. Thus, UE1 can only defer the new RAC request to after the RAC timeout at *t*+10. This increases the RTT for the data segment.

Then, we gradually boost the sending rate by increasing the size of the TCP receive window. Figure 5 shows the average RTT. The latter begins to grow with the window size, as TCP data segments experience higher queueing time at UE1's RLC buffer. However, we observe that the RTT *decreases* after a certain point when the *data* flow is IM. This is because UE1 approaches a *full buffer* condition. In this case, almost each UL data transmission transports a trailing non-zero BSR, hence UE1 no longer needs to send RAC requests to obtain a grant, hence the RTT is shorter. On the other hand, when the data flow is in DM, every new segment requires a RAC request, since trailing BSRs may not be overheard by the eNB: this is because the DM transmission power may be too low for them to get to the eNB, or because the eNB may reuse frequency on a spatial basis, allowing other DM pairs to use the same RBs, thus being unable to overhear single transmissions. Note that, with DM-IM, the ACK flow is never large enough to approach a full-buffer condition, hence RAC requests are still required for that. Thus, DM for the data flow is inefficient at higher sending rates.

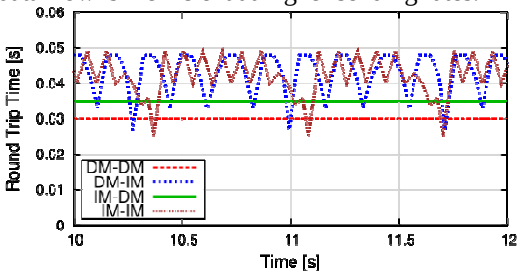

Figure 3 - Evolution of RTT over time

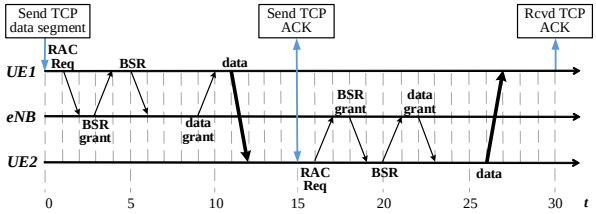

Figure 4 - Analysis of RTT for DM-DM communications

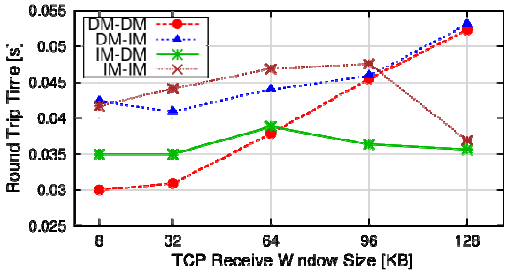

Figure 5 - RTT with different TCP receive window size

IV. Dynamic D2D connections

We now evaluate the performance of a TCP connection between two D2D-capable UEs in mobility. Mobility causes changes in the link quality, so it is desirable that the communication switches from DM to IM and vice versa. Mode switching may cause relevant losses. In fact, a single hop in LTE-A (both the SL one and either leg of the IM path) has its own PDCP peering, with associated state (e.g., PDU numbering) and ciphering. When the mode is switched, the traffic buffered below the PDCP layer (e.g., at the RLC) for the "old" mode cannot be sent on the new path, and the UE can only drop it. TCP is sensitive to losses, which it interprets as congestion signals, thereby reducing the congestion window (*cwnd*), hence the throughput. Well-known congestion control algorithms react differently when a(n alleged) congestion is perceived. In particular, mode switching

represents a case study where a possibly large burst of data segments is lost and several duplicate ACKs (*dupACKs*) reach the TCP sender almost simultaneously. Figure 2, right, reports the simulation scenario, which is designed on purpose to highlight the impact of mode switches. An omnidirectional eNB has two D2D-capable UEs associated to it. These start at a distance of 300m to the eNB, and move back and forth along a line at 3m/s, so that their distance varies in [30m; 160m]. This way, each UE has a constant UL CQI equal to 9, and a varying CQI on the SL, going both above and below 9. Every second, the eNB selects and enforces the highest-CQI mode, hence the communication bounces between DM and IM. Fading and inter-cell interference are disabled, and we verified that residual physical-layer errors are negligible and do not affect our results. The traffic is the same as Section III, i.e. a 1GB-file transfer from UE1 to UE2. The receive window size is 128 KB. Figures 6a-f represent the evolution of the cwnd for different TCP implementations. For conciseness, we focus on a single mode switch event occurring at about *t*=221s (solid arrow in the figures), but the same behavior occurs several times in a run, since the communication periodically bounces from DM-DM to IM-IM and vice versa.

a) Reno [6]: it implements Fast Retransmit and Fast Recovery (FRec), thus UE1 halves its cwnd once it receives three or more *dupACKs* and enters the recovery phase, where it retransmits one segment per RTT. During this phase, UE1 increases the cwnd by the number of dupACKs received (*ndup*) until an ACK for new data is received, then it exits the recovery phase (this means reducing the cwnd by *ndup*). However, the unACKed segments are still in flight from UE1's point of view, thus the number of outstanding segments exceeds the sending window, preventing UE1 from transmitting the next segment until the retransmit timeout expires after 1s. UE1 resumes transmission from the first unACKed segment, but it ends retransmitting also those segments correctly received by UE2, thus generating another congestion event (the dotted arrow in the figure).

b) NewReno [7]: same as Reno but the sender exits the FRec phase only when *all* outstanding segments at the congestion event have been ACKed (i.e., reception of a *full* ACK). During FRec, every *partial* ACK i) reduces the cwnd by the number of ACKed segments and ii) increases the cwnd by one. This yields the plateau in Figure 6b. Note that, one second after the congestion event, the cwnd is reset to 1 MSS. In fact, the retransmit timer is restarted only after the reception of the *first* partial ACK, which expires before the end of the FRec phase and causes the cwnd to be reset to 1 MSS without exiting FRec (i.e. UE1 continues to send one segment per RTT). This is called the *impatient variant* of NewReno: once FRec terminates, UE1 switches to slow-start, instead of congestion-avoidance, to allow faster recovery. NewReno prevents UE1 from stalling, although its FRec phase may last several seconds.

c) Westwood [8]: same as Reno but instead of halving the cwnd, the latter is set according to the estimation of the end-to-end bandwidth. The latter is calculated by measuring the rate of the returning ACKs. The rationale is to avoid an excessive reduction of the cwnd when the congestion signal is instead due to temporary link failures (e.g. errors on a wireless link). However, Figure 6c shows that cwnd is reduced to a very low value, thus failing its goal, due to a phenomenon known as *ACK compression*

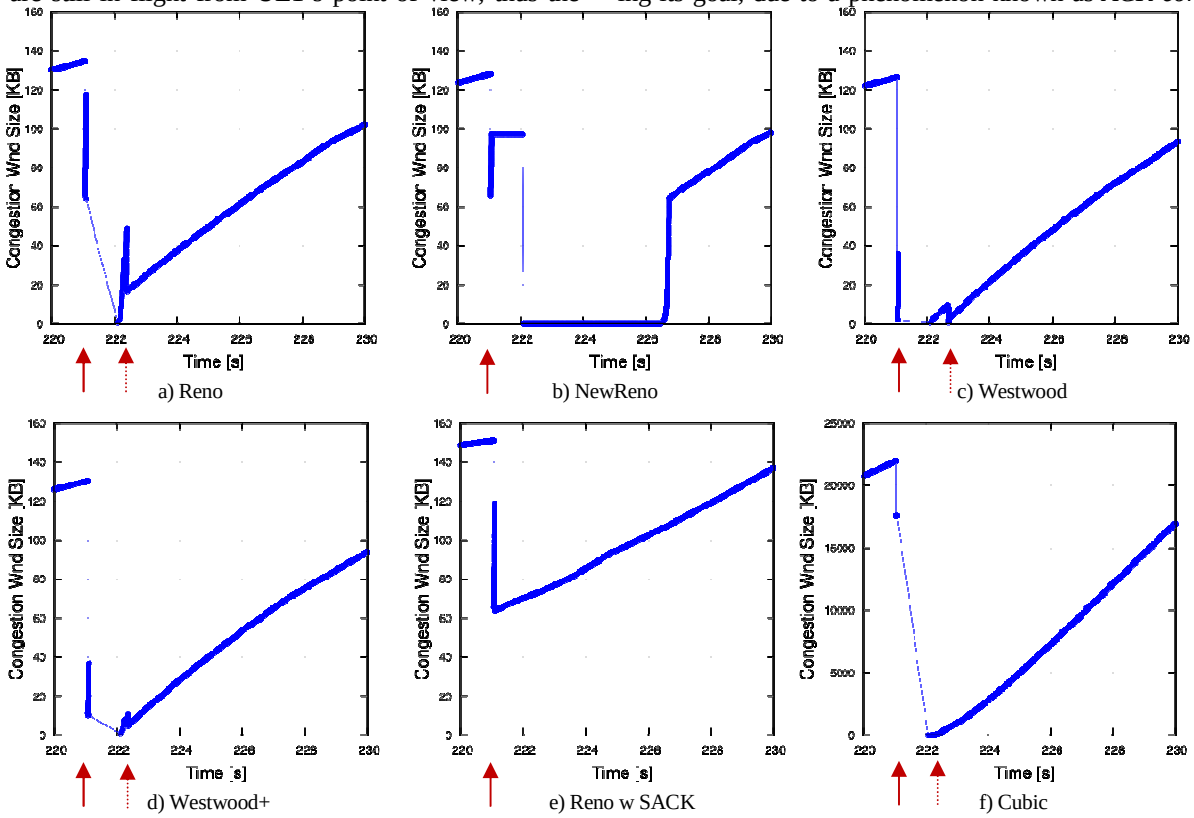

Figure 6 – Evolution of the congestion window over time with different TCP congestion control algorithms.

[9]. Each bandwidth sample is computed as $b_k = d_k / \Delta_k$, where d_k is the amount of data ACKed and Δ_k is the time distance between two consecutive ACKs. Since in LTE-A the time is slotted, two ACKs may arrive at the same time, hence $\Delta_k = 0$, resulting in undefined behavior. In this implementation, b_k is set to 0, thus strongly affecting the bandwidth estimation. Moreover, Westwood presents the same problems as Reno.

d) Westwood+ [10]: a variant of Westwood where term Δ_k is set to the value of the last RTT to avoid ACK compression. However, the cwnd still undergoes a coarse reduction to about 13 KB and exhibits a similar behavior as with Reno.

e) Reno with Selective ACKs [11]: it uses the same algorithm of Reno to increase/decrease the cwnd, but adds the SACK option to the TCP segment. Each returning ACK contains the indication about a set of out-of-sequence data received at UE2. This avoids both i) to wait for a retransmit timer after congestion and ii) to trigger a second congestion event. In fact, selective ACKs allows UE1 to know exactly the number of outstanding segments and to avoid deadlocks. On the other hand, already received segments are not retransmitted by UE1, thus dupACKs are not generated. Thus, the second congestion does not occur.

f) Cubic [12]: the cwnd growth function is cubic and does not depend on the RTT (note the larger scale of the *y* axis). At congestion, UE1 reduces cwnd by a factor β (typically 0.8) and retransmits one segment. However, as in Reno, the unACKed segments exceed the sending window, thus UE1 has to stop and wait for the retransmit timeout to expire. In this case too, a second congestion event is triggered. Since the growth function is cubic, the cwnd reverts to a value comparable to the receive window faster than the other cases.

Figure 7 shows the application-layer throughput obtained with the different congestion control algorithms. The mode switches do affect the throughput, thus Reno with Selective ACKs and Cubic outperform the other implementations. The former does not drastically reduce its cwnd, whereas the latter recovers very fast. Reno, Westwood and Westwood+ perform almost the same, as they all present the same problems. On the other hand, NewReno's smaller throughput is due to the longer duration of the recovery phase at mode switch.

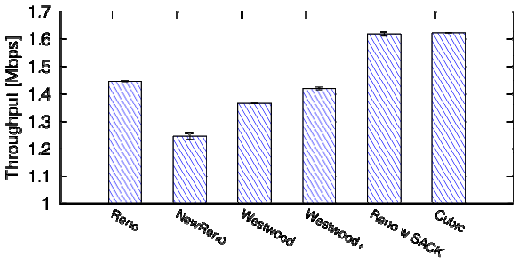

Figure 7 – Application-layer throughput

V. CONCLUSIONS

We evaluated the performance of TCP-based applications over D2D communications. We showed that the RTT is affected by protocol mechanisms within the LTE-A stack, like RAC and bandwidth stealing. Simulation results showed that D2D communication allows moderate reduction of the RTT. Surprisingly, traditional infrastructure communications may even achieve better performance when the sending rate of the TCP flow is high. Moreover, we showed that multiple losses occurring after D2D mode switching cause undesired behaviors when using some TCP implementations, which treat losses as congestion signal. Results showed that using TCP Cubic or selective acks reduces the impact of mode switching.

REFERENCES

[1] 3GPP - TS 36.843 v12.0.1, "Study on LTE Device to Device Proximity Services: Radio aspects (Release 12)", March 2014.
[2] SimuLTE webpage. http://www.simulte.com.
[3] A. Virdis, G. Stea, G. Nardini, "Simulating LTE/LTE-Advanced Networks with SimuLTE", doi: 10.1007/978-3-319-26470-7_5, Advances in Intelligent Systems and Computing, vol.402, pp.83-105, Springer, 2016.
[4] OMNeT++, http://www.omnetpp.org
[5] V. Jacobson, "Congestion avoidance and control," Proc. of SIGCOMM Symp. on comm. architectures and protocols, pp.314-329, 1988.
[6] V. Jacobson, "Modified TCP congestion avoidance algorithm," Technical report, 30 Apr 1990. Email to the end2end-interest mailing list.
[7] S. Floyd, T. Henderson, "The NewReno modification to TCP's Fast Recovery mechanism," RFC 2582, April 1999.
[8] C. Casetti, M. Gerla, S. Mascolo, M. Sanadadi, R. Wang, "TCP Westwood: End-to-end congestion control for wired/wireless networks," Wireless Networks, September 2002, vol.8, issue 5, pp.467-479.
[9] J.C. Mogul, "Observing TCP dynamics in real networks," Proc. of ACM SIGCOMM 1992, pp.305-317.
[10] L. Grieco , S. Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control," ACM SIGCOMM Computer Communication Review, v.34 n.2, April 2004.
[11] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective acknowledgement options," RFC 2018, April 1996.
[12] S.Ha, I. Rhee, L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," ACM SIGOPS Operating Sys. Rev., vol.42(5), pp.64-74, July 2008.
[13] 3GPP TR 36.814 v9.0.0, "Further advancements for E-UTRA physical layer aspects (Release 9)," March 2010.
[14] K. Doppler, Chia-Hao Yu, C.B. Ribeiro, P. Janis, "Mode Selection for Device-To-Device Communication Underlaying an LTE-Advanced Network," Proc. of WCNC 2010, pp.1-6, 18-21 Apr. 2010.
[15] J. Gu, S.J. Bae, B.G. Choi, M.Y. Chung, "Mode Selection Scheme Considering Transmission Power for Improving Performance of Device-to-Device Communication in Cellular Networks", Proc. ICUIMC 2012, 20-22 Feb. 2012, Kuala Lumpur, Malaysia.
[16] M. Belleschi, G. Fodor, A. Abrardo, "Performance analysis of a distributed resource allocation scheme for D2D communications," IEEE GLOBECOM 2011 workshops, pp.358-362, 5-9 Dec. 2011.
[17] X. Xiao, X. Tao, J. Lu, "A QoS-Aware Power Optimization Scheme in OFDMA Systems with Integrated Device-to-Device (D2D) Communications," Proc. of VTC Fall 2011 pp.1-5, 5-8 Sept. 2011.
[18] S. Wen, X. Zhu, X. Zhang, D. Yang, "QoS-aware mode selection and resource allocation scheme for Device-to-Device (D2D) communication in cellular networks," Proc. of ICC 2013, pp.101-105, 9-13 June 2013.
[19] C. H. Yu, O. Tirkkonen, K. Doppler, C. Ribeiro, "On the Performance of Device-to-Device Underlay Communication with Simple Power Control," Proc. of IEEE VTC Spring 2009, pp. 1-5, 26-29 Apr. 2009.
[20] X. Lin, *et al.*, "An overview of 3GPP device-to-device proximity services," IEEE Comm. Mag., vol.52(4), pp.40-48, 2014
[21] G. Stea, A. Virdis, "A comprehensive simulation analysis of LTE discontinuous reception (DRX)," Computer Networks, vol. 73(2014), pp.22–40.
[22] J. Huang, *et al.* "An in-depth study of LTE: effect of network protocol and application behavior on performance," Proc. of ACM SIGCOMM 2013, pp.363-374.
[23] B. Nguyen, *et al.*, "Towards understanding TCP performance on LTE/EPC mobile networks," Proc. of the 4th workshop on all things cellular: operations, applications, & challenges. ACM, 2014, pp. 41–46.
[24] D. Pacifico, *et al.*, "Improving TCP performance during the Intra LTE handover," IEEE GLOBECOM 2009.